AVF Control Number: AVF-VSR-611.1095
15 December 1995
95-06-02-INT

Ada COMPILER
VALIDATION SUMMARY REPORT:
Certificate Number: 951017W2.0-001
Intermetrics Inc.
AdaMagic, Version 2.0
SPARCstation 5 under SUNOS, 4.1.4 =>
Raytheon Extended Weapons Control Computer (EWCC) (Bare Machine)

VSR Status: (Final)

Prepared By:
ADA VALIDATION FACILITY
88 CG/SCTL
Wright-Patterson AFB OH 45433-5707

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	15 December 1995	Final

4. TITLE AND SUBTITLE:
Ada Compiler Validation Summary Report: Intermetrics, Inc., VC# 951017W2.0-001 -- AdaMagic, Version 2.0

5. FUNDING NUMBERS

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Ada Validation Facility
88 CG/SCTL
Wright-Patterson AFB, OH 45433-5707

8. PERFORMING ORGANIZATION REPORT NUMBER
AVF-VSR-611.1095

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Ada Joint Program Office, Department of Defense
Code JEXSV, 701 S. Courthouse Rd.
Arlington, VA 22204-2199

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
Host: SPARCstation 5 (under SunOS 4.1.4)
Target: Raytheon Extended Weapons Control Computer (EWCC) (bare machine)

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; Distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 words)***
This report documents the validation testing of an Ada 95 implementation. This testing was conducted according to the Ada Compiler Validation Procedures, Version 4.0, using Ada Compiler Validation Capability test suite version 2.0, and completed 17 October 1995.

14. SUBJECT TERMS
Ada programming language, Ada Compiler Validation Capability, Ada Compiler Validation Summary Report, Ada Joint Program Office, Ada Validation Organization, Ada Validation Facility, testing, ISO/IEC 8652:1995

15. NUMBER OF PAGES
42

16. PRICE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED

NSN 7540-01-280-5500

TABLE OF CONTENTS

Preface

Validation Certificate

Declaration of Conformance

CHAPTER 1 INTRODUCTION

1.1 USE OF THIS VALIDATION SUMMARY REPORT 1-1
1.2 REFERENCES. 1-2
1.3 ACVC TEST CLASSES 1-2
1.4 SPECIAL CONDITIONS FOR ACVC 2.0 1-3
1.5 DEFINITION OF TERMS 1-3

CHAPTER 2 IMPLEMENTATION DEPENDENCIES

2.1 WITHDRAWN TESTS 2-1
2.2 INAPPLICABLE TESTS. 2-1
2.3 MODIFICATIONS 2-3
2.4 UNSUPPORTED ADA 95 FEATURES 2-6
2.4.1 Real-Time 2-7
2.4.2 OOP 2-7
2.4.3 Type Extensions in Child Units. 2-7
2.4.4 Child Library Units 2-7
2.4.5 Pre-Defined Language Environments 2-7
2.4.6 Mixed Features. 2-7

CHAPTER 3 PROCESSING INFORMATION

3.1 TESTING ENVIRONMENT 3-1
3.2 TEST EXECUTION. 3-1

APPENDIX A MACRO PARAMETERS AND IMPLEMENTATION-SPECIFIC VALUES

A.1 MACRO PARAMETERS A-1
A.2 VALUES FROM PACKAGE IMPDEF A-3

APPENDIX B COMPILATION SYSTEM OPTIONS AND LINKER OPTIONS

APPENDIX C WITHDRAWN TESTS LIST

PREFACE

This report documents the validation testing of an Ada 95 implementation. This testing was conducted according to the Ada Compiler Validation Procedures version 4.0, using the Ada Compiler Validation Capability test suite version 2.0, and completed 17 October 1995.

The successful completion of validation testing is the basis for the Ada certification body's issuance of a validation certificate and for subsequent registration of derived implementations. A copy of the validation certificate and its attachment that were awarded for this validation are presented in the following two pages. Validation testing does not ensure that an implementation has no nonconformities to the Ada 95 standard other than those, if any, documented in this report. The compiler vendor declares that the tested implementation contains no deliberate deviation from the Ada 95 standard other than the omission of features; a copy of this Declaration of Conformance is presented immediately after the certificate pages.

This report has been reviewed and approved by the signatories below. These organizations attest that, to the best of their knowledge, this report is accurate and complete; however, they make no warrant, expressed or implied, that omissions or errors have not occurred.

Brian P. Andrews

Ada Validation Facility
Brian P. Andrews
AVF Manager
88 CG/SCTL
Wright-Patterson AFB OH 45433-5707

for

Ada Validation Organization
Director, Computer and Software Engineering Division
Institute for Defense Analyses
Alexandria VA 22311 USA

Ada Joint Program Office
Dr. Charles B. Engle
Center for Information Management
Defense Information Systems Agency
Alexandria VA 22041 USA

The **Ada Joint Program Office** *awards this*

Ada Validation Certificate

No. 951017W2.0-001

to **Intermetrics, Inc.**
for successfully validating the following Ada implementation:

Compiler: AdaMagic, Version 2.0
Host Computer: SPARCstation 5 (under SunOS 4.1.4)
Target Computer: Raytheon Extended Weapons Control Computer (EWCC) (bare machine)

Conformity testing to ANSI/ISO/IEC-8652:1995 was conducted by Wright-Patterson Air Force Base, USA, using ACVC Version 2.0. This testing is documented in the AVF's Validation Summary Report #AVF-VSR-611.

The validated status conveyed by this certificate is in effect from the validation date listed below.

Validation Date: 17 October 1995
Expiration Date: 31 March 1998

AJPO

UNITED STATES DEPARTMENT OF DEFENSE,
Ada JOINT PROGRAM OFFICE

Dr. Charles B. Engle, Chief of the AJPO

Core Ada 95 Test Categories and Specialized Needs Annexes For Intermetrics; AdaMagic, Version 2.0; VC#951017W2.0-001; ACVC 2.0.

Core Ada 95 Test Categories

- **Ada 9X Basic:** This is the subset of tests from ACVC 1.11 after removal of tests not applicable to Ada 95. These tests focus on support expected from Ada for features of Ada 83 that have been updated to be compatible with revised rules.

Note: The following subsets of tests validate features that are new to Ada 95. Each test has been allocated into exactly one of several test subsets, based upon a general categorization of Ada features used in the test. These tests are designed to reflect the features that programmers are likely to use to solve a programming problem.

- **Real-Time:** This subset is composed of tests for the new Ada 95 features from Section 9: Tasks and Synchronization. These features include protected objects, modifications to task types, select statements, and delay alternatives.
- **OOP:** This subset of tests focuses on some necessary facilities for achieving object-oriented programming in Ada 95. Features validated include tagged types, class attributes, and abstract types and subprograms. Other Ada 95 facilities commonly used in object-oriented programs are included in subsequent subsets.
- **Type Extensions in Child Units:** Tests that focus on the interaction of the two new Ada features of type extensions of tagged types and child library units. This includes the related semantics of visibility, accessibility, and calls on primitive operations of tagged types.
- **Child Library Units:** Tests that focus on the support for the new Ada capability to provide a hierarchical organization of the compilation units of an Ada program with the associated capabilities of granting access to the contents of private declarations and of hiding selected units within subsystems.
- **Pre-defined Language Environment:** This subset of tests include some Ada 83 facilities and some new features defined in Annex A. Annex A provides specifications for root library units for Ada, Interface, and System, character and string handling and input/output.
- **Mixed Features:** This relatively large subset of tests focuses on the interaction of Ada features that are a mixture of familiar Ada 83 and new Ada 95 features.

Ada 95 Test Categories	Passed	Not Applicable	Not Supported	With-drawn	TOTALS
Ada 9X Basic	2472	286	0	197	2955
Real-Time	48	0	0	3	51
OOP	40	0	0	15	55
Type Extensions in Child Units	27	0	0	8	35
Child Library Units	29	0	0	8	37
Pre-defined Language Environments	10	0	0	17	27
Mixed Features	123	24	0	42	189
TOTALS	2749	310	0	290	3349

Specialized Needs Annexes

Note: Tests allocated to these annexes are processed only when vendor claims support.

Specialized Needs Annexes	Passed	Not Applicable	Not Supported	With-drawn	TOTALS
Annex C System Programming	4	0	0	3	7
Annex D Real-Time Systems	19	1	0	11	31
Annex E Distributed Systems	0	0	7	1	8
Annex F Information Systems	0	0	2	4	6
Annex G Numerics	0	0	1	4	5
Annex H Safety and Security	0	0	0	0	0
TOTALS	23	1	10	23	57

Declaration of Conformance

Customer: Intermetrics

Ada Validation Facility: 88 CG/SCTL
Wright-Patterson AFB OH 45433-5707

ACVC Version: 2.0

Ada Implementation:

Ada Compiler Name and Version: AdaMagic, Version 2.0

Host Computer System: SPARCstation 5
SUNOS, 4.1.4

Target Computer System: Raytheon Extended Weapons Control
Computer (EWCC) (Bare Machine)

Declaration:

I, the undersigned, declare that I have no knowledge
of deliberate deviations from the Ada Language Standard
(ANSI/ISO/IEC 8652:1995, FIPS PUB 119-1) other that the
ommission of features as documented in the Validation Summary Report.

Customer Signature

10/16/95
Date

CHAPTER 1

INTRODUCTION

The Ada implementation described above was tested according to the Ada Validation Procedures [Pro95] against the Ada Standard [Ada95] using the Ada Compiler Validation Capability (ACVC) Version 2.0. This Validation Summary Report (VSR) gives an account of the testing of this Ada implementation. For any technical terms used in this report, the reader is referred to [Pro95]. A detailed description of the ACVC may be found in the current ACVC User's Guide [UG95].

1.1 USE OF THIS VALIDATION SUMMARY REPORT

Consistent with the national laws of the originating country, the Ada Certification Body may make full and free public disclosure of this report. In the United States, this is provided in accordance with the "Freedom of Information Act" (5 U.S.C. 552). The results of this validation apply only to the computers, operating systems, and compiler versions identified in this report.

While the organizations represented on the signature page of this report have exercised diligent care to make this report accurate and complete, they cannot warrant that all statements set forth in this report are accurate and complete, or that the subject implementation has no nonconformities to the Ada Standard other than those presented. Copies of this report are available to the public from the AVF that performed this validation or from:

National Technical Information Service
5285 Port Royal Road
Springfield VA 22161, USA

Questions regarding this report or the validation test results should be directed to the AVF which performed this validation or to:

Ada Validation Organization
Computer and Software Engineering Division
Institute for Defense Analyses
1801 North Beauregard Street
Alexandria VA 22311-1772, USA

1.2 REFERENCES

[Ada95] Reference Manual for the Ada Programming Language, ANSI/ISO/IEC 8652:1995.

[Pro95] Ada Compiler Validation Procedures, Version 4.0, Ada Joint Program Office, January 1995.

[UG95] The Ada Compiler Validation Capability User's Guide, Version 2.0, SAIC, 6 February 1995.

1.3 ACVC TEST CLASSES

Compliance of Ada implementations is tested by means of the ACVC. The ACVC contains a collection of test programs structured into six test classes: A, B, C, D, E, and L. The first letter of a test name identifies the class to which it belongs. Class A, C, D, and E tests are executable. Class B and class L tests are expected to produce errors at compile time and link time, respectively.

The executable tests are written in a self-checking manner and produce a PASSED, FAILED, or NOT APPLICABLE message indicating the result when they are executed. Three Ada library units, the packages REPORT and SPPRT13,and the procedure CHECK_FILE are used for this purpose. The package REPORT also provides a set of identity functions used to defeat some compiler optimizations allowed by the Ada Standard that would circumvent a test objective. The package SPPRT13 contains constants of type SYSTEM.ADDRESS. These constants are used by selected chapter 13 tests and by isolated tests for other chapters. The procedure CHECK_FILE is used to check the contents of text files written by some of the Class C tests for the Input-Output features of the Ada Standard, defined in Annex A of [Ada 95]. The operation of REPORT and CHECK_FILE is checked by a set of executable tests. If these units are not operating correctly, validation testing is discontinued.

Class B tests check that a compiler detects illegal language usage. Class B tests are not executable. Each test in this class is compiled and the resulting compilation listing is examined to verify that all violations of the Ada Standard are detected. Some of the class B tests contain legal Ada code which must not be flagged illegal by the compiler. This behavior is also verified.

Class L tests check that an Ada implementation correctly detects violation of the Ada Standard involving multiple, separately compiled units. Errors are expected at link time, and execution is attempted.

For some tests of the ACVC, certain implmentation-specific values must be supplied. The insertion method for the implementation specific values can either be a macro substitution on the source file level of the test or linking a package that contains the implementation specific values. Details are described in [UG95]. A list of the values used for this implementation

is provided in Appendix A. In addition to these anticipated test modifications, additional changes may be required to remove unforeseen conflicts between the tests and implementation-dependent characteristics. The modifications required for this implementation are described in section 2.3.

For each Ada implementation, a customized test suite is produced by the AVF. This customization consists of making the modifications described in the preceding paragraph, removing withdrawn tests (see section 2.1), and possibly removing some inapplicable tests (see section 2.2 and [UG95]).

1.4 SPECIAL CONDITIONS FOR ACVC 2.0

ACVC 2.0 is designated to cover the transition period from Ada 83 to Ada 95. For details see [Pro95]. ACVC 2.0 is composed of two sets of tests. The first set is commonly called Ada 9X Basic. It consists of tests that were taken from ACVC 1.12 with possibly minor modifications to remove incompatibilities with Ada 95. The second set of tests was developed in order to test truly new features of Ada 95. A consequence of this approach is that the naming conventions for tests are not uniform. The test name of a test from ACVC 9X Basic always refers to the Ada 83 Standard, even if the feature covered by the test was moved to a different chapter in [Ada95].

[Pro95] determines that the minimum requirement for passing ACVC 2.0 is to pass ACVC 9X Basic.

1.5 DEFINITION OF TERMS

Acceptable result	A result that is explicitly allowed by the grading criteria of the test program for a grading of passed or inapplicable.
Ada compiler	The software and any needed hardware that have to be added to a given host and target computer system to allow transformation of Ada programs into executable form and execution thereof.
Ada Compiler Validation Capability (ACVC)	The means for testing compliance of Ada implementations, consisting of the test suite, the support programs, the ACVC user's guide and the template for the validation summary report.
Ada Implementation	An Ada compilation system, including any required runtime support software, together with its host computer system and its target computer system.
Ada Joint Program Office (AJPO)	The part of the certification body which provides policy and guidance for the Ada certification system.

Ada Validation Facility (AVF)	The part of the certification body which carries out the procedures required to establish the compliance of an Ada implementation.
Ada Validation Organization (AVO)	The part of the certification body that provides technical guidance for operations of the Ada certification system.
ARM 83	The Ada Reference Manual, published as ANSI/MIL-STD-1815A-1983 and ISO 8652-1987. Citations from the ARM take the form <section>.<subsection>:<paragraph>.
Certification Body	The organizations (AJPO, AVO, AVFs), collectively responsible for defining and implementing Ada validation policy, including production and maintenance of the ACVC tests, and awarding of Ada validation certificates.
Compliance of an Ada Implementation	The ability of the implementation to pass an ACVC version.
Computer System	A functional unit, consisting of one or more computers and associated software, that uses common storage for all or part of a program and also for all or part of the data necessary for the execution of the program; executes user-written or user-designated programs; performs user-designated data manipulation, including arithmetic operations and logic operations; and that can execute programs that modify themselves during execution. A computer system may be a stand-alone unit or may consist of several inter-connected units.
Conformity	Fulfillment by a product, process or service of all requirements specified.
Customer	An individual or corporate entity who enters into an agreement with an AVF which specifies the terms and conditions for AVF services (of any kind) to be performed.
Declaration of Conformance	A formal statement from a customer assuring that conformity is realized or is attainable on the Ada implementation for which validation status is realized.
Foundation Unit	An Ada package used by multiple tests. Foundation units are designed to be reusable. A valid foundation unit must be in Ada library for those tests that are dependent on the foundation unit.
Host Computer System	A computer system where Ada source programs are transformed into executable form.

Inapplicable Test	A test that contains one or more test objectives found to be irrelevant for the given Ada implementation.
ISO	International Organization for Standardization.
Operating System	Software that controls the execution of programs and that provides services such as resource allocation, scheduling, input/output control, and data management.
Special Needs Annex	One of annexes C through H of [Ada95]. Validation against one or more special needs annexes is optional. For each annex, there is a test set that applies to it. In addition to all core language tests, the appropriate set of tests must be processed according to the Ada language for an implementation to be validated for a special needs annex.
Target Computer System	A computer system where the executable form of Ada programs are executed.
Validated Ada Compiler	The compiler of a validated Ada implementation.
Validated Ada Implementation	An Ada implementation that has been validated successfully either by AVF testing or by registration [Pro95].
Validation	The process of checking the conformity of an Ada compiler to the Ada programming language and of issuing a certificate for this implementation.
Withdrawn Test	A test found to be incorrect and not used in conformity testing. A test may be incorrect because it has an invalid test objective, fails to meet its test objective, or contains erroneous or illegal use of the Ada programming language.

CHAPTER 2

IMPLEMENTATION DEPENDENCIES

2.1 WITHDRAWN TESTS

At the time of this validation testing, 313 tests were withdrawn from the ACVC 2.0 test suite; these tests were not processed. Appendix C contains the withdrawn tests list, which identifies each withdrawn test and the reason(s) for its withdrawal.

2.2 INAPPLICABLE TESTS

A test is inapplicable if it contains test objectives which are irrelevant for a given Ada implementation. Reasons for a test's inapplicability may be supported by documents issued by the ISO and the AJPO known as Ada Commentaries and commonly referenced in the format AI-ddddd. For this implementation, the following tests were determined to be inapplicable for the reasons indicated; references to Ada Commentaries are included as appropriate.

The following 17 tests check for the predefined type SHORT_INTEGER; for this implementation, there is no such type:

B36105C	C45231B	C45304B	C45411B	C45502B
C45503B	C45504B	C45504E	C45611B	C45613B
C45614B	C45631B	C45632B	B52004E	C55B07B
B55B09D	CD7101E			

The following 16 tests check for the predefined type LONG_INTEGER; for this implementation, there is no such type:

C45231C	C45304C	C45411C	C45502C	C45503C
C45504C	C45504F	C45611C	C45613C	C45614C
C45631C	C45632C	B52004D	C55B07A	B55B09C
CD7101F				

B23003F checks that maximum line length and maximum identifier length are the same. For this implementation, there is no maximum (other than storage limits) and no errors are detected. (See section 2.3.)

C45231D and CD7101G check for a predefined integer type with a name other than INTEGER, LONG_INTEGER, or SHORT_INTEGER; for this implementation, there is no such type.

C45322A and C45622A check that the proper exception is raised if MACHINE_OVERFLOWS is TRUE and the results of various floating-point operations lie outside the range of the base type; for this implementation, MACHINE_OVERFLOWS is FALSE.

C45531I..P (8 tests), C45532I..P (8 tests), and CE3804J check fixed-point operations for types that require a SYSTEM.MAX_MANTISSA of 25 or greater; for this implementation, MAX_MANTISSA is less than 25. (See section 2.3.)

C96005B uses values of type DURATION's base type that are outside the range of type DURATION; for this implementation, the ranges are the same.

CE2103A, CE2103B, and CE3107A use an illegal file name in an attempt to create a file and expect NAME_ERROR to be raised; this implementation does not support external files and so raises USE_ERROR. (See section 2.3.)

The following 241 tests check operations on sequential, text, direct access, wide_text, and stream files; this implementation does not support external files:

CE2102A..C (3)	CE2102G..H (2)	CE2102K	CE2102N..Y (12)
CE2103C..D (2)	CE2104A..D (4)	CE2106A..B (2)	CE2108E..H (4)
CE2109A..C (3)	CE2110A	CE2110C	CE2111A..C (3)
CE2111E..G (3)	CE2111I	CE2120A..B (2)	CE2201A..N (14)
CE2203A	CE2204A..D (4)	CE2205A	CE2206A
CE2208B	CE2401A..C (3)	CE2401E..F (2)	CE2401H..L (5)
CE2403A	CE2404A..B (2)	CE2405B	CE2406A
CE2407A..B (2)	CE2408A..B (2)	CE2409A..B (2)	CE2410A..B (2)
CE2411A	CE3102A..B (2)	CE3102F..H (3)	CE3102J..K (2)
CE3103A	CE3104A..C (3)	CE3106A..B (2)	CE3107B
CE3108A..B (2)	CE3110A	CE3112C..D (2)	CE3114A
CE3115A	CE3119A	EE3203A	EE3204A
CE3207A	CE3301A	CE3302A	CE3304A
CE3305A	CE3401A	CE3402A	EE3402B
CE3402C..D (2)	CE3403A..B (2)	CE3403E..F (2)	CE3404B..D (3)
CE3405A	CE3405C..D (2)	CE3406A..D (4)	CE3407A..C (3)
CE3408A..C (3)	CE3409A	CE3409C..E (3)	EE3409F
CE3410A	CE3410C	CE3410E	CE3411A
CE3411C	CE3412A	EE3412C	CE3413A..C (3)
CE3414A	CE3602A..D (4)	CE3603A	CE3604A..B (2)
CE3605A..E (5)	CE3606A..B (2)	CE3704A..B (2)	CE3704D..F (3)

CE3704M..O (3)	CE3705A..E (5)	CE3706D	CE3706F..G (2)
CE3804A..E (5)	CE3804G..I (3)	CE3804M	CE3804O
CE3805A..B (2)	CE3806A..B (2)	CE3806D..E (2)	CE3806G..H (2)
CE3902B	CE3904A..B (2)	CE3905A..C (3)	CE3905L
CE3906A..C (3)	CE3906E..F (2)	CXA8001..3 (3)	CXAA001..9 (9)
CXAA011..5 (5)	CXAB001	CXAC003	CXACA01
CXACB01			

CE3704C, CE3706C, CE3804F, CE3804P, CE3806C, CE3806F, and CE3906D contain the expression "FIELD'LAST + 1". This expression is rejected at compile time by this implementation because it is a static expression whose value exceeds the base range of the type. (See section 2.3.)

CXB3002, CXB3003, and CXB5001 check if packages Interfaces.C.Strings, Interfaces.C.Pointers, and Interfaces.Fortran are available for use. This implementation does not support C of Fortran interfaces. (See section 2.3.)

CXD2003 checks that when Task_Dispatching_Policy is FIFO_Within_Priorities and a blocked task becomes ready, it is added to the tail of the ready queue for its priority. This system is a multiprocessor system and can not ensure that the low priority tasks could be started in parallel.

2.3 MODIFICATIONS

In order to comply with the test objective it may be required to modify the test source code, the test processing method, or the test evaluation method. Modifications are allowable because at the time of test writing not all possible execution environments of the test and the capabilities of the compiler could be foreseen. Possible kinds of modification are

- Test Modification: The source code of the test is changed.
  Examples for test modifications are the insertion of a pragma, the insertion of a representation clause, or the splitting of a B-test into several individual tests, if the compiler does not detect all intended errors in the original test.

- Processing Modification: The processing of the test by the Ada implementation for validation is changed.
  Examples for processing modification is the change of the compilation order for a test that consists of multiple compilations or the additional compilation of a specific support unit in the library.

- Evaluation modification: The evaluation of a test result is changed.
  An example for evaluation modification is the grading of a test other than the output from REPORT.RESULT indicates. This may be required if the test makes assumptions about implementation features that are not supported by the implementation (e.g. the implementation of a file system on a bare target machine).

Modifications were required for 64 tests.

The following tests were split into two or more tests because this implementation did not report the violations of the Ada Standard in the way expected by the original tests.

b23004a	b24204d	b38003c	b38009d	b44004c
b55a01a	b83e01c	b83e01d	b83e01e	ba1101c
bc1109a	bc1109b	bc1109c	bc1109d	bc2001d
bc51017				

B23003F was graded inapplicable by Evaluation Modification as directed by the AVO. This test nominally checks that the maximum line & identifier lengths are equal. But there is no such requirement in [Ada 95]; rather, [Ada 95] 2.2(15) requires implementations to support identifiers and lines of at least 200 characters. Thus, this test is retained to check that a maximum identifier length is enforced. However, for this implementation there is no fixed maximum line or identifier length; the implementation ensures that identifiers of 200 characters are always accepted, but varies the maximum length according to remaining room in the lexer's buffer. The AVO ruled that this was acceptable for validation under ACVC 2.0 pending a ruling from the ARG.

B38105A, B45601A, B54A05B, B87B48C were graded passed by Test Modification as directed by the AVO. These tests include the assignments of fixed-point values that lie outside of the range of the fixed-point base type because this implementation uses all extra bits for precision. These tests were modified to use values that were within range, as follows:

for B38105A at line 80 '1.0' was changed to '2.0' (widening the range);
for B45601A at line 33, B54A05B at line 33, and B87B48C at line 38
1.0 was changed to 0.0

C3A2001, C954A01..3, CA11004/5/9/16/19, & CC54001..4 (13 tests) were graded passed by Test Modification as directed by the AVO. These tests depend on an elaboration order that is not required by [Ada 95], so in order to avoid the raising of Program_Error, an Elaborate or Elaborate_Body pragma was inserted at the following places:

```
C3A2001 after line 366   for (Report, TCTouch, C3A2001_1, C3A2001_2,
                              C3A2001_3, C3A2001_4);
C954A01 at    line  90   for (F954A00);
C954A02 at    line  90   for (F954A00);
C954A03 at    line  98   for (F954A00);
CA11004 at    line f0-58 with Elaborate_Body;
CA11005 at    line f0-49 with Elaborate_Body;
CA11009 at    line  51   with Elaborate_Body;
CA11016 at    line 195   for (CA11016_0.CA11016_1);
CA11019 at    line 192   for (CA11019_0.CA11019_1);
CC54001 after line  98   for (CC54001_0);
CC54002 after line 101   for (CC54002_0);
CC54003 after line 162   for (CC54003_0);
CC54004 at    line 208   for (CC54004_2)
```

C43003A was graded passed by Test Modification as directed by the AVO. This test includes a fixed-point subtype declaration with an upper bound that is outside of the range of this implementation's base type because all extra bits are used for precision. The upper bound was modified, as follows:

at line 224 change '8.0' to '7.75'

C45531I..L, C45532I..L, & CE3804J (9 tests) were graded inapplicable as directed by the AVO. These tests use fixed-point types that require more than 24 bits, which exceeds this implementation's capacity. (CE3804J would be inapplicable in any case for this implementation, since it requires support of external files.)

C95040D was graded passed by Evaluation Modification as directed by the AVO. This test attempts I/O from separate tasks to the same file object, which results in interleaved output text.

EA3004G was graded passed by Evaluation Modification as directed by the AVO. This test expects the reference to an obsolete unit to be detected at compile time; this implementation makes the detection at link time.

CA1020D was graded passed by Processing Modification as directed by the AVO. This test checks that certain previously compiled library units in a compil- ation are made obsolete by a later compilation. This implementation makes all units of a compilation obsolete if any one is, as allowed by [Ada 95] 10.1(4), and so the needed units in file CA1020D0 are not available as expected by the test. This test was split into 10 separate compilations as shown below by duplicating files CA1020D0 & -D1 and commenting out various lines so as to isolate the units expected to be made obsolete), and then all 10 files (including -D2 and -D3M) were processed:

```
for CA1020D0
(a) comment lines 51..65          (leaves generic units)
(b) comment lines 32..49, 55..65 (leaves CA1020D_Proc1)
(c) comment lines 32..53, 59..65 (leaves CA1020D_Func2)
(d) comment lines 32..57, 63..65 (leaves CA1020D_Proc3)
(e) comment lines 32..61          (leaves CA1020D_Func3)

for CA1020D1
(a) comment lines 45..56          (leaves CA1020D_Proc1 & CA1020D_Func2)
(b) comment lines 33..42, 53..56 (leaves CA1020D_Proc3)
(c) comment lines 33..53          (leaves CA1020D_Func3)
```

CA5004B was graded passed by Processing Modification as directed by the AVO. This test checks that an Elaborate pragma is obeyed when it is given for a unit whose body has yet to be compiled or is replaced. However, this implementation doesn't permit a compilation to contain units with the same name. The test was split into 3 separate compilations as follows:

(a) lines 1..59; (b) lines 60..80; & (c) lines 81..175

BC1002A was graded passed by Evaluation Modification as directed by the AVO. This test includes a fixed-point type declaration that requires more than 24 bits, which exceeds this implementation's capacity and is rejected by the compiler. This additional error citation was ignored.

CE2103A, CE2103B, and CE3107A were graded inapplicable by Evaluation Modification as directed by the AVO. The tests abort with an unhandled exception when USE_ERROR is raised on the attempt to create an external file. This is acceptable behavior because this implementation does not support external files (cf. AI-00332).

CE3704C, CE3706C, CE3804F/P, CE3806C/F, & CE3906D (7 tests) were graded inapplicable by Evaluation Modification as directed by the AVO. Thest tests use the static expression "FIELD'LAST + 1" which is rejected at compile time because it exceeds the range of the base type. (These tests would otherwise execute and report "not applicable" because this implementation doesn't support external files.)

CE3810B was graded passed by Test Modification as directed by the AVO. This test checks Fixed_IO and uses a fixed-point type that requires more than 24 bits which exceeds this implementation's capacity. The type declaration at line 44 was changed to use '250.0' vice '1000.0' as the range's upper bound.

CXB3002, CXB3003, & CXB5001 were graded inapplicable by Evaluation Modification as directed by the AVO. These tests check the support of interfaces to (foreign) languages C and Fortran. This implementation doesn't support such interfaces, as allowed by [Ada 95] B.2(11,13).

CXD1002 was graded passed by Processing Modification as directed by the AVO. This test checks that the base priority of the main subprogram can be set by a Priority pragma. This implementation requires a special option to designate a user-defined library unit as the main subprogram; this test was processed with the linker option "-pru" designating this test.

2.4 UNSUPPORTED ADA 95 FEATURES

As allowed by [Pro95], not all tests for Ada 95 features need be passed for validation under ACVC 2.0; this allowance is intended to facilitate the development of quality implementations during the transition period. Furthermore, support of the [Ada95] Special Needs Annexes is always an implementation option, and partial support might be appropriate for some application domains. The relevant tests have been categorized as described by the attachment to the validation certificate, shown at the front of this report; the identification of the complete set of tests for each category and each Annex is given in [UG95]. For this implementation, the following tests were graded as "not supported".

2.4.1 Real-Time

No REAL-TIME tests were graded "not supported".

2.4.2 OOP

No OOP tests were graded "not supported".

2.4.3 Type Extensions in Child Units

No tests with type extensions in child units were graded "not supported".

2.4.4 Child Library Units

No child library unit tests were graded "not supported".

2.4.5 Pre-Defined Language Environments

No tests of pre-defined language environments were graded "not supported".

2.4.6 Mixed Features

No tests of mixed features were graded "not supported".

CHAPTER 3

PROCESSING INFORMATION

3.1 TESTING ENVIRONMENT

The Ada implementation tested in this validation effort is described adequately by the information given in the initial pages of this report.

For technical and sales information about this Ada implementation, contact:

Mike Ryer
Intermetrics Inc.
733 Concord Ave
Cambridge MA 02138
(617) 661-1840

Testing of this Ada implementation was conducted at the {customer's | AVF's} site by a validation team from the AVF.

3.2 TEST EXECUTION

A magnetic tape containing the customized test suite (see section 1.3) was taken on-site by the validation team for processing. The contents of the magnetic tape were loaded onto the host computer.

The tests were compiled and linked on the host computer system, as appropriate. The executable images were transferred to a DECStation 5000/200 system by a magnetic tape and loaded onto the target computer via Adaview, and run.

Testing was performed using command scripts provided by the customer and reviewed by the validation team. See Appendix B for a complete listing of the processing options for this implementation. It also indicates the default options. The options invoked explicitly for validation testing during this test were:

compilation option	meaning
-nz	The nz flag non-zeros the heap for detecting errors more easily.
-lc or -lp	When listings are required, the -lc flag creates a continuous (non-paginated) source (and error) listing. The -lp flag creates a paginated source (and error) listing.

linker option	meaning
-nc	("no recompile"). This disables an automatic recompilation feature; use of this feature is sometimes not consistent with the expected testing results.
-alt	Use "alternate" elaboration routine. The release of the compiler that we checkpointed assumes a Raytheon usage. The elaboration order is different if we run stand alone or on top of the TD/DSK.
-ll	The linker options are passed from adabuild. What gets passed is -c acvc.lc. The acvc.lc contains the locating segment information for the hardware.
-ol	Passes the hardware version of the time_support to the linker. The load file from the compiler has a dummy ewcc_time_support body which needs to be replaced at link time either with the simulator version or the harware one. We replace it with the hardware version.

Test output, compiler and linker listings, and job logs were captured on magnetic tape and archived at the AVF. The listings examined on-site by the validation team were also archived.

APPENDIX A

MACRO PARAMETERS AND IMPLEMENTATION-SPECIFIC VALUES

This appendix contains the macro parameters used for customizing the ACVC. The meaning and purpose of these parameters are explained in [UG95]. The parameter values are presented in two tables. The first table lists the values that are defined in terms of the maximum input-line length, which is the value for $MAX_IN_LEN, also listed here. These values are expressed in a symbolic notation, using placeholders as appropriate.

A.1 MACRO PARAMETERS

Macro Parameter	Macro Value
$MAX_IN_LEN	200
$BIG_ID1	AAA ... A1 (<$MAX_IN_LEN> characters)
$BIG_ID2	AAA ... A2 (<$MAX_IN_LEN> characters)
$BIG_ID3	AAA ... A3A ... A (<$MAX_IN_LEN> characters)
$BIG_ID4	AAA ... A4A ... A (<$MAX_IN_LEN> characters)
$BIG_STRING1	"AAA ... A" (<$MAX_IN_LEN>/2 characters)
$BIG_STRING2	"AAA ... A1" ((<$MAX_IN_LEN>/2)-1 characters)
$BLANKS	" " (<$MAX_IN_LEN>-20 blanks)
$MAX_STRING_LITERAL	"AAA ... A" (<$MAX_IN_LEN> characters)
$ACC_SIZE	24
$ALIGNMENT	1
$COUNT_LAST	8388607

MACRO PARAMETERS AND IMPLEMENTATION-SPECIFIC VALUES

```
$ENTRY_ADDRESS                  FCNDECL.To_Address (16#9700C#)
$ENTRY_ADDRESS1                 FCNDECL.To_Address (16#9700D#)
$ENTRY_ADDRESS2                 FCNDECL.To_Address (16#9700E#)
$FIELD_LAST                     8388607
$FORM_STRING                    ""
$FORM_STRING2                   "CANNOT_RESTRICT_FILE_CAPACITY"
$GREATER_THAN_DURATION          86_401.0
$GREATER_THAN_DURATION_BASE_LAST
                                131_073.0
$GREATER_THAN_FLOAT_BASE_LAST   1.80141E+38
$GREATER_THAN_FLOAT_SAFE_LARGE  1.0E308
$ILLEGAL_EXTERNAL_FILE_NAME1    \NODIRECTORY\FILENAME
$ILLEGAL_EXTERNAL_FILE_NAME2    THIS-FILE-NAME-IS-TOO-LONG-FOR-MY-SYSTEM
$INAPPROPRIATE_LINE_LENGTH      -1
$INAPPROPRIATE_PAGE_LENGTH      -1
$INTEGER_FIRST                  -8388608
$INTEGER_LAST                   8388607
$INTEGER_LAST_PLUS_1            8388608
$INTERFACE_LANGUAGE             NO_LANGUAGE_AVAILABLE
$LESS_THAN_DURATION             -90_000.0
$LESS_THAN_DURATION_BASE_FIRST  -131_073.0
$MACHINE_CODE_STATEMENT         RP'(OTHERS => 0);
$MAX_INT                        8388607
$MAX_INT_PLUS_1                 8388608
$MIN_INT                        -8388608
$NAME                           NO_SUCH_TYPE_AVAILABLE
$NAME_SPECIFICATION1            X2120A
$NAME_SPECIFICATION2            X2102B
```

$NAME_SPECIFICATION3	X2120C
$OPTIONAL_DISC	(blank character)
$RECORD_DEFINITION	RECORD DUMMY : INTEGER; END RECORD;
$RECORD_NAME	RP
$TASK_SIZE	48
$TASK_STORAGE_SIZE	2048
$VARIABLE_ADDRESS	FCNDECL.To_Address(16#97000#)
$VARIABLE_ADDRESS1	FCNDECL.To_Address(16#97001#)
$VARIABLE_ADDRESS2	FCNDECL.To_Address(16#97002#)

A.2 VALUES FROM PACKAGE IMPDEF

```
-- IMPSPEC.ADA

   Minimum_Task_Switch : constant duration := 0.1;
   --                                         ^^^ --- VALIDATION VALUE
   -- This is the minimum time required to allow another task to get
   -- control.  It is expected that the task is on the Ready queue.
   -- A duration of 0.0 would normally be sufficient but some number
   -- greater than that is expected.
   --

   Switch_To_New_Task : constant duration := 0.5;   -- change for ewcc target
   --                                         ^^^ -- VALIDATION VALUE
   -- This is the time required to activate another task and allow it
   -- to run to its first accept statement.
   --

   Clear_Ready_Queue : constant duration := 5.0;
   --                                       ^^^ --- VALIDATION VALUE
   -- This is the time which will clear the queues of other tasks
   -- waiting to run.  It is expected that this will be about five
   -- times greater than Switch_To_New_Task.
   --

   Delay_For_Time_Past : constant duration := 0.1;
   --                                         ^^^ --- VALIDATION VALUE
   -- Some implementations will boot with the time set to 1901/1/1/0.0
   -- This constant is such that when a delay of Delay_For_Time_Past is given,
   -- the implementation guarantees that a subsequent call to
   -- Ada.Calendar.Time_Of(1901,1,1) will yeild a time that has already passed
   -- (for example, when used in a delay_until statement)
```

```
--

Procedure Exceed_Time_Slice;
-- This is a procedure who's execution is guaranteed to be greater
-- than the time slice unit on implementations which use time slicing
-- For those which do not implement time slicing this could be
-- null;
--

type Processor_Type is (Uni_Processor, Time_Slice, Multi_Processor);
--
Processor : constant Processor_Type := Multi_Processor;  -- changed for ewcc
--                                     ^^^^^^^^^^ --- VALIDATION VALUE
-- Indicates the type of processor on which the tests are running.
--    Time_Slice indicates a uni-processor with an operating system
--    that simulates a multi-processor by using time slicing

Interrupt_To_Use_For_Testing : constant Ada.Interrupts.Interrupt_ID
  := Ada.Interrupts.Interrupt_ID'First;  -- to allow trivial compilation
--   ^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^^  --- VALIDATION VALUE
A_Reasonable_Amount_Of_Time_To_Wait_For_An_Interrupt : constant := 10.0;
--                                                  VALIDATION VALUE --- ^^^^
-- These two constants are used for interrupt testing in the
-- Systems Annex.  If the Systems Annex tests are not going to be
-- used, it is allowed to delete these two constants, along with the
-- above reference to Ada.Interrupts.Names;
--
-- Interrupt_To_Use_For_Testing should be set to the name of an
-- interrupt ID that will ideally cause interrupts within the time
-- interval specified by:
-- A_Reasonable_Amount_Of_Time_To_Wait_For_An_Interrupt.

function Negative_Zero return Float;
-- This function must return a "negative zero" value
-- for implementations for which Float'Signed_Zeros is True.
--
```

APPENDIX B

COMPILATION SYSTEM OPTIONS AND LINKER OPTIONS

The compiler options of this Ada implementation, as described in this Appendix, are provided by the customer. Unless specifically noted otherwise, references in this appendix are to compiler documentation and not to this report.

```
adacomp 1.694 (ALPHA):
     Copyright (c) 1994, 1995, Intermetrics, Inc.  All Rights Reserved.
Usage: adacomp [options ...] [file ...]

    file                A source file to be compiled.  There may be
                        multiple files specified.

Options Summary:

Listing Options

    -lc                   Continuous source listing interspersed with messages.
    -lp                   Paginated source listing interspersed with messages.
    -lr                   Relevant-only source listing, (only source lines
                          for which there are error or warning messages).
    -le                   Source listing only if there are errors.
    -lx                   Cross reference listing  (turns on -xr).
    -pl length            Set page length of source listing file to length.
    -pw width             Set page width of source listing file to width.
    -rl                   Record layout listing for all record types.
    -prl                  Record layout listing for packed record types only.

Message Options

    -m  msg_kind          Suppresses the display of any messages of msg_kind
                          for the current invocation of the compiler.
    +m  msg_kind          Enables the display of any messages of msg_kind
                          for the current invocation of the compiler.
    -mr msg_kind          Suppresses the display of any messages of msg_kind
                          for any recursive invocations of the compiler.
    +mr msg_kind          Enables the display of any messages of msg_kind
                          for any recursive invocations of the compiler.

    The valid values for msg_kind:
```

```
        a - all messages
        d - implementation-dependent warning messages
        e - error messages
        i - information messages
        n - nyi messages
        w - general warning messages
        r - redundant messages
```

By default, all messages except information and redundant messages are displayed. For recursive invocations, no messages are displayed by default. For convenience, "-m a" will suppress all messages *except* errors.

Miscellaneous Options

```
-a                  Analyzer only, don't run the emitter,
                  backend or optimizer.
-asm                Save the interleaved assembler listing.
-c                  Frontend only, don't run the backend or optimizer.
-e count            Stop reporting errors after the count but keep on going.
-f                  Force generation of il even if there are errors.
-g                  Generate information for symbolic debugger.
-help or -h         Display this help message.
-s                  Suppress all checks.
-N                  Suppress certain numeric checks.
-O level            Call the optimizer with optimizing level:
                  all, debug or none.
                    The default is to have optimization or all.
                    -g debug disables optimizations which substantially
                    interfere with debugging.
-t                  Trace each declaration and statement passed to
                  the emitter.
-nr                 No "current heap" releases (just keep allocating,
                    never releasing).
-nz                 Initialize all heap memory to a non-zero value.
                    (In hex, the non-zero value is BAD1BAD1 so it
                    is easy to spot in the debugger, and causes a
                    Bus Error on the Sparc when dereferenced.)
-xr                 Save xref info for the Browser.
```

Driver Options

```
-0                  Identifies executable version numbers (default).
-of file            Read options from specified file.
-ke                 Keep intermediate files.
-ki                 Keep the info file .
-ne                 Don't re-exec adacomp process on failure.
-nl                 Don't re-exec adacomp process on last file.
-pB "BE options"
-pO "Optimizer options"
-pL "Lister options"
                  Pass options to Back End, Optimizer or Lister.
                  (Multiple options must be enclosed in quotes.)
                  Debugging toggles for phases can also be passed here.
-q                  Quiet mode -- suppress all inessential messages.
```

```
    -sr                 (Search and Register) Enable automatic registration of
                      source files
    -T                  Print timing information on compiler phases.
    -v                  Verbose mode -- driver reports its every action.
```

Executable File Overrides

```
    -xd dir_path        Override default ADA_MAGIC environment
                      variable if exists.
    -xB exe_path        Override default back end.
    -xL exe_path        Override default lister.
    -xO exe_path        Override default optimizer.
```

(See manual for more details.)

Linker Options

The linker options of this Ada implementation, as described in this Appendix, are provided by the customer. Unless specifically noted otherwise, references in this appendix are to linker documentation and not to this report.

```
adabuild 1.694 (ALPHA):
     Copyright (c) 1994, 1995, Intermetrics, Inc.  All Rights Reserved.
Usage: adabuild [options ...] [unit ...]

    unit                The main unit to be linked.  There may be
                        multiple units specified.

Options Summary:

    -help or -h             Display this help message.
    -v                      Provide verbose output.
    -td file            For dual linker, pass "file" to llink from
                          import library.
    -ol file            Pass ol file "file" to llink.
    -pru unit           Use certain pragmas of "unit" to override
                            main unit pragmas.
    -dh number              Pass "-dh number" directly to llink.
                        "-dh" is a required llink option.  If there is
                        no "-dh" specified to adabuild, a default
                        "-dh 262144" is passed to llink
    -ll option              Pass "option" to llink.
                        To pass multiword options, repeat "-ll", i.e.,
                        to pass "-c foo" use "-ll -c -ll foo".
    -g                      Build with debugging symbols.
    -na                     No autoregistration.
    -nc                     No recompilations.
    -no                     No ".ol out of date" recompilations.
    -nl                     No link (prelink, but do not call llink).
    -nse                No llink segment elimination (seg elim is the default).
    -ke                 Keep intermediate files.
    -f                      Force linking, despite any prelinker errors.
    -alt                Use "alternative" elaboration routine (at_alt.ol)
```

APPENDIX C

WITHDRAWN TESTS LIST

B23003D + imposes line-/lexical_element-length rules not required by Ada95

B23003E + imposes line-/lexical_element-length rules not required by Ada95

B32104A : Ada95 3.3.1:9 permits unconstrained array definitions for objects

B32201A : @69-85 Ada95 allows non-universal initial values for named numbers

B34003B : @160,177 'Pred,'Succ are defined for floating-point in Ada95

B34004B : @116,120,158,162,179 the fixed-point operations are legal in Ada95

B34005N : @181 requires K to be universal_integer; Ada95 allows any integer

B34005T : @229 requires K to be universal_integer; Ada95 allows any integer

B34007H : @178 requires K to be universal_integer; Ada95 allows any integer

B34007K : @206 'STORAGE_SIZE is legal in Ada95 by implicit dereference

B34007Q : @151 'CONSTRAINED is legal in Ada95 by an implicit dereference

B34007T : @161 'CONSTRAINED is legal in Ada95 by an implicit dereference

B34008B : @143 applies 'Address to a derived task type, which isn't a prog.unit

B34014D : @81-2,130-1,174-5 is resp. illegal/ok by 3.2.3:7 & 3.4:17

B34014F : @81-2,130-1,174-5 is resp. illegal/ok by 3.2.3:7 & 3.4:17

B34014M : @77-8,112-3,151-2,187-8,228-9,266-7 are resp. illegal/ok by 3.4:17

B34014Q : @82/3,135/6,183/4,194/5,237/8,277/8 is resp. illegal/ok by 3.4:17

B34014S : @82/3,135/6,186/7 is resp. illegal/ok by 3.2.3:7 & 3.4:17

B34014Z : @78/9,115/6,156/7,196/7,237/8,277/8 is resp. illegal/ok by 3.4:17

WITHDRAWN TESTS LIST

B35401A : @40,41 is legal in Ada95 (array bounds can be static)

B35501A : @55,63 'SUCC,'PRED are defined for fixed-point types in Ada95

B36101A : @79,110 'A'..'Z' is ambiguous in Ada95 (CHARACTER vs. WIDE_CHARACTER)

B36171A : @120,128 declares an object w/unconstr.array def.--legal in Ada95

B36201A : @112,113 'First(N) only needs a static integer N--each is, in Ada95

B37004A : @76 is legal since this doesn't cause freezing (8.6:17 & 13.14:8)

B37401A : @178,179,182 'CONSTRAINED is allowed by Ada95 (func.call,impl.deref.)

B38103A : @175,184,212,248,257,261,268 checks Ada83 errors now legal by 6.3.1

B38103B : @179,188,216,252,261,265,272 checks Ada83 errors now legal by 6.3.1

B38103C : @f2-31,40,68,104,113,117,124 Ada83 errors are legal by Ada95 6.3.1

B38103D : @186,195,223,259,268,272,279 Ada83 errors are legal by Ada95 6.3.1

B38103E : @f1-31,40,68,104,113,117,124 Ada83 errors are legal by Ada95 6.3.1

B391001 : all but the last ERROR are illegal also by accessibility rules

B392002 : @141,144 violates 3.9.2:12 (dispatching operation of two types)

B392004 : @176,178,180 violates 3.9.2:9 as each call is dynamicly tagged

B392006 : @104,108,115 is legal, as the function calls are tag indeterminate
: @113 (comment re 112) is wrong, operand is controlling, like others

B392007 : @113 has a known disc.part but parent type is unconstrained (3.7:13)

B393001 : @145 inherits but doesn't override Return_Vis_Abstract (3.9.3:6)

B393003 : @197 is legal, the procedure overrides an inherited operation
: @214 the subprogram overrides no homograph, so is legal
: @242 is legal, the procedure overrides an inherited operation
: @248 the subprogram overrides no homograph, so is legal

B393004 : @127,134,149 the functions must be declared abstract (3.9.3:8'last)
: @184,210,241 Process & Update aren't defined for New_Field (3.2.3:6)
: @78,79 the comment is wrong to imply inheritance of Process & Update

B3A0001 : @108 violates accessibility rule 3.10.2:28
: @243 is legal (the access, not designated, value is updated)

B3A2002 : @204 is legal (3.10:9), an aliased view
: @215,217 the attribute 'Access violates 3.10.2:27 [extra error @217]

B3A2003 : @220,221 'Access prefix must be variable, not constant (3.10.2:25)

B3A2007 : @ all but one instantiation violates accessibility rule 3.9.1:3

B48002F : @44 the allocator applies a constraint to an elementary type (4.8:4)

B54A08A : @42,47 a case expression may be of a generic formal type (5.4:4)

B54A10A : @57,62 are legal in Ada95, which uses any integer vs. Universal_Int

B55B14B : all checks are invalid, as 'Range is static in these cases in Ada95

B63009A : @165,174,202,240,251,255,262,274,277 are legal by Ada95 6.3.1

B63009B : @169,178,206,244,255,259,266,278,281 are legal by Ada95 6.3.1

B63009C : @f1-31,40,68,106,117,121,128,140,143 are legal by Ada95 6.3.1

B63102A : the test objective is invalid by Ada95 6.3.1

B66001D : @42 has a static value out of range of the base type (4.9:35)

B731A01 : @157 is legal (private operations are declared & visible here)
: @78 the comment is wrong re visibility & hence legality

B731A02 : @169 is legal (private operations are declared & visible here)

B74103B : the Ada83 errors are legal in Ada95 (private'SIZE parameter default)

B74103C : all of the intended Ada83 errors are legal in Ada95

B74103F : many of the intended Ada83 errors are legal in Ada95

B74103H : all intended Ada83 errors ERRORs are legal by 13.14:8 (no freezing)

B74104A : @174,183,211,247,256,260,267 Ada83 errors are legal by Ada95 6.3.1

B74105A : @51,55,78 Ada83 errors are legal in Ada95 (definite subtype)

B74205B : all intended errors are legal in Ada95 (implicit dereference)

B74207A : uses undefined numeric attributes & 'BASE for composite types

B74303A : @46,47 is too restrictive re deferred constants (7.4:6,4.9.1:2)

B74304B : @49,50,59,60 Ada83 errors are legal by Ada95 freezing rules

B74304C : @54 the Ada83 error is legal by Ada95 freezing rules

B74409A : @68 overloads "=" which is legal in Ada95

B83011A : @137-139 checks an Ada83 visibility rule changed in Ada95 (8.2:2)

B83041E : @86,88 "*" & "/" resolve to universal_fixed (4.5.5:18-20)

B85001F : @52 Ada83 error is legal in Ada95 (renaming function result object)

WITHDRAWN TESTS LIST

B85013C : @58,87 Ada95 allows an others choice

B95020A : @124,172,178,198,234,237,242,250,252 meets Ada95 conformance rules

B95020B : @f2-50,56,76,104,112,115,120,128,130 meets Ada95 conformance rules

B95074E : has an invalid objective for Ada95 (implicit dereference)

BA1010B : @f3-32 profile conforms in Ada95 (explicit/implicit mode "in")

BA1010F : @f2-32 profile conforms in Ada95 (explicit/implicit mode "in")

BA1010G : @f1-32 profile conforms in Ada95 (explicit/implicit mode "in")

BA1010H : @f1-32 profile conforms in Ada95 (explicit/implicit mode "in")

BA1010I : @f2-32 profile conforms in Ada95 (explicit/implicit mode "in")

BA1010J : @f3-36 profile conforms in Ada95 (explicit/implicit mode "in")

BA1010M : @f2-36 profile conforms in Ada95 (explicit/implicit mode "in")

BA1010N : @f1-36 profile conforms in Ada95 (explicit/implicit mode "in")

BA1010P : @f1-36 profile conforms in Ada95 (explicit/implicit mode "in")

BA1010Q : @f2-36 profile conforms in Ada95 (explicit/implicit mode "in")

BA11003 : @112 generic package BA11003_7 cannot have a body
: @129,142,148 renamed or instantiated units are not visible

BA11005 : @117 has an illegal (not required) package body

BA11006 : all ERRORs are legal, since BA11006_0._1._2 is a private descendent

BA11007 : @174 does not declare an overriding procedure, and is legal
: @234 the "/" visible is of the wrong type (no use type clause)

BA11008 : @132,150 has illegal (not required) package bodies
: @139 is legal, not an ERROR
: @187 the instantiation isn't in the parent's decl.region--10.1.1:18
: @231 has an unintended error--the pkg. name violates 6.1:8

BA11009 : @200 the body for BA11009_4.BA11001_6 is illegal (not required)

BA12003 : @158,160 the context clauses are illegal (10.1.2:8)

BA12005 : @134,136 violates 10.1.2:8, as _0._1._4 is considered a declaration
: @36ff (diagram) shows a unit BA12005_7 that's not in the test

BA12007 : @206 a package cannot rename a function
: @313 should have a context clause for BA12007_0._1._3 (function)
: @318 obj.decl lacks subtype_ind. before func.initialization express.

BA2011A : @f0-56,62,f2-56,70,f3-76,f4-33,51 conforms in Ada95 (6.3.1)

BA3001C : @28 the package body is illegal in Ada95 (there must be no body)

BC1226A : @61,63,78,79 is too restrictive re deferred constants (7.4:6,4.9.1:2)

BC3202A : @64,73 subtype NREC doesn't staticly match (12.5.1:14)

BC3202B : @63 subtype NREC doesn't staticly match (12.5.1:14)

BC3202D : @91,100 subtype REC7 doesn't staticly match (12.5.1:14)

BC3205C : all package bodies in this test must not be provided in Ada95

BC3220B : checks a restriction on use of "others" that Ada95 lacks (4.3.3:11)

BC3403A : @97,109,110,121 violates Ada95 generic parameter-matching rules

BC3403B : @90,102,107,108 violates Ada95 generic parameter-matching rules

BC3404C : @53 violates Ada95 generic parameter-matching rules

BC3404D : @77 violates Ada95 generic parameter-matching rules

BC3502A : @42 violates Ada95 generic parameter-matching rules

BC3502F : @57,64,65 violates Ada95 generic parameter-matching rules

BC3502G : @66 the instantiation requires a definite subtype

BC3502H : @103-111 R3,R8 subtypes don't staticly match (12.5.1:14)

BC51007 : @206,210,214,218 an extension's accessibility level exceeds parent's

BC51014 : @86 library package BC51014_0 is not visible (missing context clause)

BC51C01 : @88,384,404 instantiating w/Abstract_Child violates 3.9.3:9 re Proc

BC54003 : @269 has an indefinite actual where the formal is definite

BD1B05E : @254 is legal (non-static default expressions don't cause freezing)

BD2A11A : a 'SIZE clause for scalar types w/non-static bounds is legal in Ada95

BD2A13B : all expected errors may be legal in Ada95 (13.1:23)

BD4008B : checks an Ada83 (13.4:7) restriction that Ada95 does not impose

BD5002A : Ada95 does not require WITH SYSTEM in order to use an address clause

BD5002F : has an invalid objective for Ada95 ("with System;" isn't needed)

BD5101A : Ada95 (J.7.1:4) allows 'Address clauses for entries w/out" param.s

WITHDRAWN TESTS LIST

BD5101B : Ada95 (J.7.1:4) allows 'Address clauses for entries w/out" param.s

BD5101C : Ada95 (J.7.1:4) allows 'Address clauses for entries w/out" param.s

BD7302A : @43,44 'MACHINE_RADIX is defined for fixed-point types in Ada95

BD8003A : Ada95 allows code statements in functions as well as in procedures

BDD2001 : @66 needs 'tagged limited' for a limited completion @87 & 'Class @283
: @306/7 the attribute representation clauses are for non-local items
: @321,331,334 assignments are made to a limited object

BXAC004 : @172,173,177,178,181 Streams is not directly visible
: @307 the record extension is not accessible from its parent type
: @337,344 assigns to a limited type

C24203B : @106/110 uses Ada83 numeric attribute 'LARGE, undefined in Ada95

C34003A : @215-266,286-303,313-316 uses undefined Ada83 attributes

C34003C : @67-70 uses 'SAFE_LARGE which is undefined in Ada95

C34004A : @236,247-259,269-281,287 uses undefined Ada83 fixed-point attributes

C34004C : uses Ada83 numeric attributes that are undefined in Ada95

C34008A : applies 'ADDRESS to a derived (task) type, not a task unit

C34014C : Z is initialized by a different function by Ada95's inheritance rules

C34014E : Z is initialized by a different function by Ada95's inheritance rules

C34014P : Z is initialized by a different function by Ada95's inheritance rules

C34014R : Z is initialized by a different function by Ada95's inheritance rules

C35505D : <integer>'SUCC (<integer>'BASE'LAST) need not raise an exception

C35904A : @104,129 static values potentially are not in the base range

C35A03C : checks Ada83 fixed-point'MANTISSA which is undefined in Ada95

C35A07N : uses Ada83 numeric attribute 'LARGE which is undefined in Ada95

C35A07Q : @93,97,142 uses Ada83 fixed-point'LARGE which is undefined in Ada95

C36105B : uses static INTEGER expressions exceeding INTEGER'LAST (4.9:35)

C37211E : @196 an allocator cannot constrain an elementary type (4.8:4)

C37213A : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C37213C : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C37213E : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C37213G : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C37214A : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C37215A : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C37215C : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C37215E : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C37215G : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C37216A : requires Ada83-3.7.2:5 subtype compatibility checks not made in Ada95

C38102E : @100 uses uninitialized variables

C390003 + @294 the 2nd parameter is set as "Veh", causing some failures
+ @331 the 2nd parameter should be "MC", not "Veh"
+ @377-382 the expected/passed tags are incorrect (3.9:25,4.6:42)

C391001 : @50 type Object requires a limited completion @61-63 (7.3:6)

C391002 : @176 "limited" is a syntax error

C392B04 : @f1-109 30.0 isn't exactly representable, thus checks @215,229 fail

C392C07 : @165 wrongly expects Switch.Create to set Toggle.On=FALSE

C393009 : @118,129 isn't mode conformant with inherited Handle @66 (out/in out)

C393010 : @71 violates 3.7:11 having a default for tagged type

C393A03 : @89 violates 3.9.3:8 by assigning to an abstract target

C393B12 : @96 violates accessibility rule 3.9.1:3

C393B13 : @52 violates 3.7:11 by having a default for tagged type

C3A0014 : @363 violates 12.5.1:9 by instantiating Gen with a constrained type

C42005A : @47 the null string literal violates 4.9:34—upper bound<'Base'First

C43004B : in Ada95, array bounds slide in a subtype conversion

C43103B : @123,142,162 expects CONSTRAINT_ERROR but in Ada95 it won't be raised

C432003 : @519 will raise Constraint_Error—array bounds 1..5 for 1..10

C43213A : in Ada95, array bounds slide in a subtype conversion

C44003E : uses user-defined fixed-point "*","/"—always ambiguous in Ada95

WITHDRAWN TESTS LIST

C452001 : @531,536 expects the wrong result for (in)equality checks
        : @343 the comment is mistaken re what result to expect

C45232A : uses literals required to exceed 'BASE'LAST which is illegal in Ada95

C45242A : uses literals required to exceed 'BASE'LAST which is illegal in Ada95

C45331A : @many checks requires the Ada83 value for fixed-point 'SMALL

C45331D : @many checks requires the Ada83 value for fixed-point 'SMALL

C45411A : uses -INTEGER'FIRST which exceeds INTEGER'LAST so is illegal in Ada95

C45411D : uses -DT1'FIRST which exceeds DT1'BASE'LAST so is illegal in Ada95

C45423A : has an obsolete objective relating to safe numbers

C45523A : @89 uses Ada83 numeric attribute 'SAFE_LARGE, undefined in Ada95

C45534A : @90 F:=1.0 exceeds FIX'BASE'LAST which is illegal in Ada95

C45651A : @222,236,255 requires the Ada83 value for <fixed>'Small

C45652A : the test objective is not met (exception is raised outside of ABS)

C45701A : @82 uses Ada83 numeric attribute 'SAFE_LARGE, undefined in Ada95

C46023A : @76,78,83 uses 'MANTISSA & 'LARGE which are undefined in Ada95

C48005C : @59,84 an allocator cannot constrain an elementary type (4.8:4)

C48008B : @85,95,105 an allocator cannot constrain an elementary type (4.8:4)

C48008D : @64,74 an allocator cannot constrain an elementary type (4.8:4)

C48009D : in Ada95, array bounds slide in a subtype conversion

C48009E : in Ada95, array bounds slide in a subtype conversion

C52012A : objective re assignment targets on exception violates 11.6:6,13.9.1:5

C52012B : objective re assignment targets on exception violates 11.6:6,13.9.1:5

C52103S : in Ada95, the null arrays' non-null dimension lengths must match

C54A13D : @63 the choice is illegal; it should occur for the 2nd case stmnt

C55B08A : checks Ada83 CONSTRAINT_ERROR conditions that are illegal in Ada95

C64105E : in Ada95, the null arrays' non-null dimension lengths must match

C64105F : in Ada95, the null arrays' non-null dimension lengths must match

C74208B : @42 freezes type REC2 hence also REC before it's completely defined

C74305A : @82 C2'FIRST freeze C2 before its full declaration (13.14:18)

C760001 + @256,258,314,317 Adjust needn't be called (7.6:13-15,21,4.3:5)

C760007 + @200,201,204,205 expects too many Adjust calls (7.6:13-15,21,4.3:5)
+ @186-190 declares a procedure (Examine) that's never used

C761003 + @127 may have elaboration problems initializing Null_Global (95-5234)
+ @167 has a context clause for Report which isn't used in the package
+ @244 Subtest_1_Expected_Inits should be 4 (Item_4 has explicit init.)
+ @263 the check can fail because @126 evaluation may call Finalize
+ @354,357,360 Sup.Validate fails because it doesn't reset Inits_Called
+ @357,360 wrongly checks an order of components' finalization (7.6:12)

C761004 + @178 Subtest_1_Expected_Inits should be 4 (Item_4 has explicit init.)
+ @185 the check can fail because @183 evaluation may call Finalize
+ @234 wrongly checks an order of components' finalization (7.6:12)

C761005 : @141 must be "limited" as the completion @153 is limited (7.3:6)

C85018B : @270 sliding occurs with no CONSTRAINT_ERROR (6.4.1:10/11 & 4.6:37)

C854001 : @250,255 parameters to "-" & Other_Name are uninitialized

C87B35B : @72-74 "*","/" are ambiguous--Ada95 allows the operand type REAL

C940A03 : @109 the declaration freezes Key_Type which isn't completely declared
: @109 initialization would require the package body to be elaborated

C95008A : @119 the range 'A'..'Y' is ambiguous in Ada95

C95086E : @165 a length check on a null array fails for a non-null dimension

C95086F : @165 a length check on a null array fails for a non-null dimension

C954020 : @290 task Credit_Computation wrongly blocks, causing failure

C954022 : @144 loops infinitely as there is no call to Sequencer.Input

C954024 : @202 the entry Start has no corresponding call, causing failure
: @295 the select guard will only be evaluated once, causing failure

C96005C : uses static values outside of DURATION'BASE'RANGE which is illegal

CA11003 : @260 an uninitialized IN parameter is used in a check

CA11006 : @194 an uninitialized IN parameter is used in a check

CA11013 : @186,187 operator /= isn't directly visible (missing use type clause)
: @126 type My_Float should have minimal precision (digits 6)

CA11014 : @140 operator = isn't directly visible (missing use type clause)

WITHDRAWN TESTS LIST

CA11018 : @181,186 might fail an elaboration check for the instantiation
        : @321,347 raises CONSTRAINT_ERROR in a call to Copy (strings mismatch)

CA13001 : @339 uses an uninitialized actual by-copy IN OUT parameter

CC1311B : @183, in Ada95, array bounds slide in a subtype conversion

CA3009A + requires an old unit to be used if a new one has errors (10.1.4:6)

CC30001 : @152,161 violates accessibility rule 3.9.1:3

CC3208A : @69,107 discriminant subtypes don't staticly match (12.5.1:14)

CC3208B : @74,112 discriminant subtypes don't staticly match (12.5.1:14)

CC3208C : @74,112 discriminant subtypes don't staticly match (12.5.1:14)

CC3406A : @56 array subtypes don't staticly match (12.5.3:6)

CC3406B : @60 array subtypes don't staticly match (12.5.3:6)

CC3406C : @65 array subtypes don't staticly match (12.5.3:6)

CC3406D : @55 array subtypes don't staticly match (12.5.3:6)

CC3407A : @68,86 the generic actual parameters don't staticly match formals

CC3407B : @68,84 generic actual parameters don't staticly match formals

CC3407C : @68,86 generic actual parameters don't staticly match formals

CC3407D : @76,95,116 generic actual parameters don't staticly match formals

CC3407E : @67 generic actual parameters don't staticly match formals

CC3407F : @56 generic actual parameters don't staticly match formals

CC3408A : @56 array subtypes don't staticly match (12.5.3:6)

CC3408B : @59 array subtypes don't staticly match (12.5.3:6)

CC3408C : @67 array subtypes don't staticly match (12.5.3:6)

CC3408D : @54 array subtypes don't staticly match (12.5.3:6)

CC3601A : instantiations @137,140,143,146,187,192,240 require definite subtypes

CC50001 : has an invalid objective--expecting the wrong "=" (3.9.2:14,15,20)

CC50A01 : @255,263 violates accessibility rule 3.9.1:3

CC51003 : @160 violates 12.5.1:9 by instantiating with a constrained type

CC51004 : @154 violates 12.5.1:9 by instantiating with a constrained type

CC51007 : @231 violates 3.9.1:3--extension declared at deeper accsblty. level

CC70001 : @176 should instantiate List_Mgr.CC70001_1 (10.1.1:18/19)

CD1C04B : @47 has a type-related representation item in violation of 13.1:10

CD1D02A : has an invalid objective; in Ada95 pragma PACK @54 is illegal

CD1D03A : has an invalid objective; in Ada95 pragma PACK @57 is illegal

CDA101B + expects that a renamed entry can access a deallocated task--unclear

CDA201B : uses Ada83 floating-point attribute'LARGE which is undefined in Ada95

CE2401D : instantiates Direct_IO with an unconstrained type, illegal in Ada95

CE2401G : instantiates Direct_IO with an unconstrained type, illegal in Ada95

CE3208A : @145 violates Ada95 A.10.3:23 (operating on a closed default file)

CE3306A : static COUNT'LAST+1 usually exceeds COUNT'BASE'LAST and is illegal

CE3402E : static COUNT'LAST+1 usually exceeds COUNT'BASE'LAST and is illegal

CE3403C : uses unqualified character literals that are ambiguous in Ada95

CE3403D : static COUNT'LAST+1 usually exceeds COUNT'BASE'LAST and is illegal

CE3409B : static COUNT'LAST+1 usually exceeds COUNT'BASE'LAST and is illegal

CE3410B : static COUNT'LAST+1 usually exceeds COUNT'BASE'LAST and is illegal

CE3410D : uses unqualified character literals that are ambiguous in Ada95

CXA3001 : @142 uses Ada.Characters.Handling.Is_Control vs. Is_Graphic

CXA3003 : @113,115,117,120 should have String upper bounds of 8,8,7,7, resp.
: @134 "is of the ISO_646" should read "is NOT of ... "
: @135 the check for non-ISO_646 characters should be " < 128 "
: @136 the Report.Failed comment should read "... IS an ISO_646 ..."

CXA4004 : the 2nd character of the string @148 should be 'b' not 'd'

CXA4005 : @116 raises CONSTRAINT_ERROR (Result_String'First - 2)
: @200,287,325,543,589 has string-matching errors
: @544 should read 'Center' vice 'Left' for Report.Failed text

CXA4008 : @182 transposes "mn" in a string check (typo)

CXA4009 : @189 "Result_String" should be "Test_String"

CXA4011 : @70,73,78,81,84,87 Unbounded_String is not directly visible

WITHDRAWN TESTS LIST

CXA4012 : @263 the 'Access prefix's accessibility level is too deep (3.10.2:32)

CXA4014 : @177 the 'Access prefix's accessibility level is too deep (3.10.2:32)

CXA4015 : @106 the 'Access prefix's accessibility level is too deep (3.10.2:32)
: @452 omitted parameter Map_Ptr and so obtains an unexpected count

CXA4016 : @120 raises CONSTRAINT_ERROR (Result_String'First - 2)
: @205,291,329,548,591 has string-matching errors
: @534 uses the undefined term 'ASW.Pad'
: @549 should read 'Center' vice 'Left' for Report.Failed text

CXA4017 : @91 an extra space in Line1 makes the check @317,318 fail

CXA4018 : @277 the parameter '"ABCDEF"' should be 'Translate("ABCDEF")'

CXA4019 : @156 the 'Access prefix's accessibility level is too deep (3.10.2:32)

CXA4020 : @227,242 "Result_String" should be "Test_String"

CXA4022 : @103,104,107,... Unbounded_Wide_String is not directly visible
: @172 the 'Access prefix's accessibility level is too deep (3.10.1:32)
: @415 Total_Count = 4, not 2

CXA4023 : @103,104,107,... Unbounded_Wide_String is not directly visible
: @165 the 'Access prefix's accessibility level is too deep (3.10.1:32)

CXA9001 : @119,171 instantiates Storage_IO with an indefinite type (12.5.1:6)

CXA9002 : @230,233,235,239,242,372,378,384,390,396 'TAG has the wrong prefix

CXAA010 : uses "Text_IO" @114, which is not visible, vs. "Ada.Text_IO"

CXAC001 : @192,193,213,214 Product_Header_* has initial, unexpected values

CXAC002 : @105 wrongly expects Name to return exactly Report.Legal_File_Name
: @158,211,217,239 the relevant "/=" operators are not directly visible

CXACA02 : @141-145 defines 'Read,'Write as 'Input,'Output--infinite recursion

CXACB02 : @323 'Input raises End_Error because the stream isn't reset
: @358 'Customer2.Customer' should be 'Customer3.Customer'

CXACC01 : @86,89,92,95,210,216,222,228 'Tag has the wrong prefix 3.9(17)

CXACC02 : @101,104,107,110 Ada.Tags isn't visible
: @101,104,107,110 'Tag has the wrong prefix, violating 3.9(17)
: @167-170 representation items are given in the wrong decl. region

CXB3001 : @121 the 2nd & 3rd parameters are of the wrong type for To_C

CXB4001 : uses a "/=" operator @210 which isn't visible (no use-type clause)

```
CXC3001 : @183 operator "/=" isn't directly visible (missing use type clause)
        : @192 CXC3001.The_Other_Handler.Count is undefined (no func.decl.)

CXC6001 : @186,190 wrongly expects Chlorine to be passed by reference (C.6:19)

CXC6002 : @202 wrongly expects CXC6002_1.Smog to be passed by reference--C.6:19

CXD1003 : @120 a Priority pragma has a value outside of Priority'Range

CXD1004 : @203 pragma priority has an out of range value
        : assumes Priority'First is near 0 (e.g., 50..100 will fail)

CXD1005 : @205 pragma priority has an out of range value
        : assumes Priority'First is near 0 (e.g., 50..100 will fail)

CXD2001 : @205 always reports FAILED for TC_Failed is never set to TRUE

CXD3001 : @76 Priority_Hi is too high--initial value exceeds subtype's range

CXD3002 : @141 pragma priority has an out of range value
        : assumes Priority'First is near 0 (e.g., 50..100 will fail)

CXD4004 : @289ff loops infinitely--Distributor & main prog. are higher priority

CXD4006 : @126,131 tasks are given the wrong priorities (swap the values)

CXD8001 : @349 expects an equality after conversion that isn't required

CXE5001 : @122 has the wrong number of parameters, but should call Do_APC
        : @119 the comment text should refer to Do_APC

CXF2001 + checks wrong values for cases 6,7,8,13,14,15,16,26,27,28,33,34,35,36

CXF3A01 : @80,131 raises Picture_Error as Valid_String(2) violates F.3.1:6

CXF3A02 : @89,141 raises Picture_Error as Valid_String(2) violates F.3.1:6

CXF3A03 : @170,etc. raises Picture_Error as Valid_String(2) violates F.3.1:6

CXG1001 : @84,166,177 operator "/=" isn't directly visible (missing use type)

CXG1003 : @172,189,287,355 "/=" isn't directly visible (missing use type)

CXG1004 : @142 generic formal parameter Real is not visible
        : has assignments to dead variables that may be omitted (11.6:5)

CXG1005 : @102,119,135,274 generic formal parameter Real is not visible
        : has assignments to dead variables that may be omitted (11.6:5)

FXF3A00 : @165 copies F.3.2:74's incorrect format string (cf. 95-5259.a)

LXD7001 : @f1-71 creates tasks dependent only on the environment task

LXD7003 : @40 misspells "No_Abort_Statements" in a Restrictions pragma
```